VBA: Centerline Editing - Address Range Split Tool

Contributed by Bert Granberg
11, May. 2009
Last Updated 12, May. 2009

This code is adapted from the HouseNumber tool that ESRI published on its ArcScripts web site. The HouseNumber tool was originally designed to work in pre-9.1 versions of ArcMap and was quite extensive in its functionality. I spoke with an ESRI staffer a few months back who indicated that, unfortunately, the code had not been adapted yet for the 9.2 reorganization of the ArcObjects component library and beyond (and there were no plans to do so). This may be why the original tool doesn't always install and run as expected.

Not wanting to install VB 6.0 to modify the HouseNumber project and compile a new DLL either, I took what I needed and adapted it to work in the VBA environment. So far so good.

What does this tool do:
- Select a street feature in an editing workspace
- With the Address Range Splitter (ARS) custom UI tool selected, click on a location on a selected street feature with the four standard address range attributes
- The existing feature will be split into two features at the point of your click and new address range attributes will be calculated using the a proportional interpolation based on feature length

Notes:
- The snapping environment IS honored although you will not see your your map tool's cursor snapping at all.
- Address Ranges are assumed to be mixed parity only. This tool will not work where mixed parity is not desired.
- The script does NOT yet take into account the address coordinate alias of intersecting streets at the click point
- The script depends on the MxDocument.OpenDocument having run since the last code interuption so if you are modifying/customizing and you get an error OR you are running the script for the first time, you'll need to run the MxDocument.OpenDocument script manually before proceeding.
- Check the results to make sure the script is producing the desired results.

How to install and configure in an ArcMap Project:

Part I. Paste code into VBA Editor

1) From the ArcMap menu, navigate to Tools --> Macros --> Visual Basic Editor to open the VBA editor

2) In the VBA Editor's Project Window, expand the Project --> Arcmap Objects folder and double click on ThisDocument

3) Paste the code (below) in the window that opens and close the VBA editor.

4) Make sure that the 3 module level variables are at the top of your code window if you have any prexisting code already in your project.

5) At the first 10 lines of the sub procedure doCenterlineSplit, modify the field names (if necessary) so they correspond to your centerline dataset's address range attribute field names.

6) Save your project

Part II. Create and configure custom ArcMap Tool

1) From the ArcMap menu, navigate to Tools -->Customize

2) Click the Commands tab.

3) Click the drop-down arrow on the Save in combo box, then change the selected item so it is the current name of your ArcMap project (ex. Untitled.mxd)

4) Click [UIControls] in the Categories list. then Click the New UIControl button.

5) Click the UIToolControl radio button and then click Create.

6) The name of the new tool appears in the Commands list. Single click the newly created UIControl, then click it again to activate in-place renamin, then type a new name for the control so it reads Project.AddressRangeSplitter.

7) Click and drag the newly created UIControl and drop it on an ArcMap toolbar or menu.

8) Keep the customize dialog box open. On the toolbar or menu you dragged the tool on to, right-click the command to set its image, caption, and other properties.

9) Save Project. Close ArcMap. Reopen the project and start using the tool.

```
'CODE 5/11/09 AGRC

'THESE 3 LINES NEED TO GO AT THE VERY TOP OF YOUR CODE WINDOW ABOVE
'PRE_EXISTING SUBS OR FUNCTIONS
Private m_pEditor As IEditor
Private m_pEditSketch As IEditSketch
Private WithEvents EditorEvents As Editor


Private Function MxDocument_opendocument() As Boolean
    Dim pUID As New UID
    pUID = "esriEditor.Editor"
    Set m_pEditor = Application.FindExtensionByCLSID(pUID)
    Set m_pEditSketch = m_pEditor
    Set EditorEvents = m_pEditor
End Function
Private Sub AddressRangeSplitter_MouseDown(ByVal button As Long, ByVal shift As Long, ByVal x As Long, ByVal y
As Long)

    Dim pDisplayTransformation As IDisplayTransformation
    Dim pScreenDisplay As IScreenDisplay
    Dim pSelFeature As IFeature
    Dim pSplitPnt As IPoint
    Dim count As Long
    Dim bol As Boolean
    Dim pMxdoc As IMxDocument
    Dim pActiveView As IActiveView
    Dim pSnapEnv As ISnapEnvironment

    Set pMxdoc = ThisDocument
    Set pActiveView = pMxdoc.ActiveView

    bol = MxDocument_opendocument()

    If Not m_pEditor.EditState = esriStateEditing Then
        MsgBox "You must currently be in an edit session to run this script.", vbOKOnly, "Exiting"
        Exit Sub
    End If

    Set pSelFeature = m_pEditor.EditSelection.Next

    If pSelFeature Is Nothing Then  ' should never be true due to enabling.
        MsgBox "The esriEditor.Editor does not have a feature selected.", vbCritical, "Can not Split"
        Exit Sub
    End If
    If Not TypeOf pSelFeature.Shape Is IPolyline Then   ' should never be true due to enabling.
        MsgBox "The Editor's selected feature is not a polyline.", vbCritical, "Can not Split"
        Exit Sub
    End If

    Set pSnapEnv = m_pEditor

    Set pSplitPnt = New Point
```

```
    Set pScreenDisplay = pActiveView.ScreenDisplay
    Set pDisplayTransformation = pScreenDisplay.DisplayTransformation
    Set pSplitPnt = pDisplayTransformation.ToMapPoint(x, y)

    'snap point to existing snap environment
    pSnapEnv.SnapPoint pSplitPnt

    Call doCenterlineSplit(pSelFeature, pSplitPnt)

    Dim pEnv As IEnvelope
    Set pEnv = pSelFeature.Extent
    pEnv.Expand 1.2, 1.2, True
    ' if the parent arc is horizontal or vertical, refresh the full screen.
    If pEnv.Width < pMxdoc.ActiveView.Extent.Width / 100 Then Set pEnv = pMxdoc.ActiveView.Extent
    If pEnv.Height < pMxdoc.ActiveView.Extent.Height / 100 Then Set pEnv = pMxdoc.ActiveView.Extent
    pActiveView.PartialRefresh 6, Nothing, pEnv

End Sub



Private Sub doCenterlineSplit(pParentFeature As IFeature, pSplitPnt As IPoint)

  On Error GoTo EH
  Dim leftFromFN As String, leftToFN As String, rightFromFN As String, rightToFN As String

  '**** Set these for each address range field
  leftFromFN = "L_F_ADD"    'LEFT FROM
  leftToFN = "L_T_ADD"      'LEFT TO
  rightFromFN = "R_F_ADD"   'RIGHT FROM
  rightToFN = "R_T_ADD"     'RIGHT TO
  '****

  Dim pParentCurve As ICurve
  Set pParentCurve = pParentFeature.Shape
  If pParentCurve Is Nothing Then ' true if user has selected a polyline with Null geometry.
    MsgBox "The selected polyline does not have a valid shape.", vbCritical, "Can not Split"
    Exit Sub
  End If
  If pParentCurve.IsEmpty = True Then
    MsgBox "The selected polyline has an empty geometry.", vbCritical, "Can not Split"
    Exit Sub
  End If
  If pParentCurve.Length = 0 Then
    MsgBox "The selected polyline has a length of zero.", vbCritical, "Can not Split"
    Exit Sub
  End If
  ' make sure the required fields which contain the house numbers have been specified.
  Dim pFields As IFields
  Set pFields = pParentFeature.Fields

  Dim leftFromFI As Long, leftToFI As Long, rightFromFI As Long, rightToFI As Long
  leftFromFI = pFields.FindField(leftFromFN)
  leftToFI = pFields.FindField(leftToFN)
  rightFromFI = pFields.FindField(rightFromFN)
  rightToFI = pFields.FindField(rightToFN)

  ' check for Null values in the 4 house number fields (modGlobals has comments on g_pExtension).
  If IsNull(pParentFeature.Value(leftFromFI)) Or _
     IsNull(pParentFeature.Value(leftToFI)) Or _
     IsNull(pParentFeature.Value(rightFromFI)) Or _
     IsNull(pParentFeature.Value(rightToFI)) Then

    MsgBox "One or more of the house numbers are Null for the selected line.", _
```

```
        vbCritical, "Can not Split"
    Exit Sub

  End If

  ' try to get a valid house number from each of the 4 house number fields.
  ' this is especially important if the fields are Text, which is common for geocoding data.
  Dim lngFrom_Left_HouseNum As Long, lngFrom_Right_HouseNum As Long, lngTo_Left_HouseNum As Long,
lngTo_Right_HouseNum As Long
  lngFrom_Left_HouseNum = TryToGetValidHouseNum(pParentFeature.Value(leftFromFI))
  lngFrom_Right_HouseNum = TryToGetValidHouseNum(pParentFeature.Value(rightFromFI))
  lngTo_Left_HouseNum = TryToGetValidHouseNum(pParentFeature.Value(leftToFI))
  lngTo_Right_HouseNum = TryToGetValidHouseNum(pParentFeature.Value(rightToFI))

  If lngFrom_Left_HouseNum = -1 Or lngFrom_Right_HouseNum = -1 Or _
     lngTo_Left_HouseNum = -1 Or lngTo_Right_HouseNum = -1 Then

    MsgBox "One or more of the house numbers are invalid for the selected line.", _
        vbCritical, "Can not Split"
    Exit Sub

  End If

  Dim mixedParity As Boolean
  mixedParity = False

  If Not mixedParity Then ' the default, test and fail if mixed parity is found.
    ' make sure both house numbers on each side of the street are either even or odd.
    Dim blnFromIsEven As Boolean, blnToIsEven As Boolean
    blnFromIsEven = isEven(lngFrom_Left_HouseNum)
    blnToIsEven = isEven(lngTo_Left_HouseNum)
    If blnFromIsEven <> blnToIsEven Then
      MsgBox "The left side of the selected line has both even and odd numbers.", _
          vbCritical, "Mixed Parity"
      Exit Sub ' fail if mixed parity.
    End If
    blnFromIsEven = isEven(lngFrom_Right_HouseNum)
    blnToIsEven = isEven(lngTo_Right_HouseNum)
    If blnFromIsEven <> blnToIsEven Then
      MsgBox "The right side of the selected line has both even and odd numbers.", _
          vbCritical, "Mixed Parity"
      Exit Sub ' fail if mixed parity.
    End If
    ' verify the 2 sides of the polyline have opposite parity (unless one side has 2 zeros).
    Dim blnLeftIsEven As Boolean
    blnLeftIsEven = isEven(lngFrom_Left_HouseNum)
    Dim blnOneSideHasZeros As Boolean
    If (lngFrom_Left_HouseNum = 0 And lngTo_Left_HouseNum = 0) Or (lngFrom_Right_HouseNum = 0 And
lngTo_Right_HouseNum = 0) Then
      blnOneSideHasZeros = True
    End If
    If blnOneSideHasZeros = False Then
      If blnLeftIsEven = blnFromIsEven Then
        If blnLeftIsEven = False Then
          MsgBox "Both sides of the selected line have odd house numbers.", vbCritical, "Mixed Parity"
        Else
          MsgBox "Both sides of the selected line have even house numbers.", vbCritical, "Mixed Parity"
        End If
        Exit Sub
      End If
    End If
  End If
  ' create the point that will be used to split the selected polyline.

  If pSplitPnt Is Nothing Then ' should never be true.
```

```
    MsgBox "A valid split point could not be created.", vbCritical, "Can not Split"
    Exit Sub
  End If
  If pSplitPnt.IsEmpty = True Then ' should never be true.
    MsgBox "A valid split point could not be created.", vbCritical, "Can not Split"
    Exit Sub
  End If
  ' split the parent feature into 2 offspring features. we use IFeatureEdit::Split since
  ' it respects geodatabase behaviour (subtypes, domains, split policies etc). it also maintains
  ' M and Z values, and works for geometric networks and topological ArcInfo coverages.
  Dim pFeatEdit As IFeatureEdit
  Set pFeatEdit = pParentFeature
  Dim pNewSet As ISet
  On Error GoTo ErrorAfterStartOperation
  m_pEditor.StartOperation
  Set pNewSet = pFeatEdit.Split(pSplitPnt)
  If pNewSet.count <> 2 Then
    MsgBox "Unknown error - two offspring lines could not be produced.", _
          vbCritical, "Can not Split"
    m_pEditor.AbortOperation
    Exit Sub
  End If
  ' we now need to modify the house numbers of the 2 offspring polylines. before doing this,
  ' we must be sure pNewFeature1 is the offspring line that contains the parent's from vertex,
  ' or our logic will not work. Since ISet::Next does not return the features in any particular
  ' order, we must test and switch if needed.
  Dim pNewFeature1 As IFeature, pNewFeature2 As IFeature
  Set pNewFeature1 = pNewSet.Next ' will be wrong 50% of the time.
  Dim pNewFeat1Curve As ICurve, pNewFeat2Curve As ICurve
  Set pNewFeat1Curve = pNewFeature1.Shape
  Dim pParentFromPnt As IPoint, pNewFeature1Pnt As IPoint
  Set pParentFromPnt = pParentCurve.FromPoint
  Set pNewFeature1Pnt = pNewFeat1Curve.FromPoint
  Dim pRelOp As IRelationalOperator
  Set pRelOp = pParentFromPnt
  If pRelOp.Equals(pNewFeature1Pnt) = True Then ' no need to switch.
    Set pNewFeature2 = pNewSet.Next
  Else ' will happen 50% of the time, need to switch.
    Set pNewFeature2 = pNewFeature1
    Set pNewFeature1 = pNewSet.Next
    Set pNewFeat1Curve = pNewFeature1.Shape
  End If
  Set pNewFeat2Curve = pNewFeature2.Shape
    Dim pNewFeature2Pnt As esriGeometry.IPoint
    Set pNewFeature2Pnt = pNewFeat2Curve.FromPoint
    'MsgBox "Parent: " & pParentCurve.SpatialReference.Name & " X = " & pParentFromPnt.x & Chr(13) & _
          "SplitPoint: " & pSplitPnt.SpatialReference.Name & " X = " & pSplitPnt.x & Chr(13) & _
          "Offspring1: " & pNewFeat1Curve.SpatialReference.Name & " X = " & pNewFeature1Pnt.x & Chr(13) & _
          "Offspring2: " & pNewFeat2Curve.SpatialReference.Name & " X = " & pNewFeature2Pnt.x
  ' get the distance along the curve (as a ratio) where the split point falls. we will need
  ' this soon for the interpolation of house numbers.
  Dim dblDistAlongCurve As Double
  dblDistAlongCurve = pNewFeat1Curve.Length / (pNewFeat1Curve.Length + pNewFeat2Curve.Length)
  ' fix the 4 house numbers that are not correct (main part of code). the other 4 numbers are
  ' already correct (the FROM_LEFT and FROM_RIGHT of the first feature, and the TO_LEFT and
  ' TO_RIGHT of the second feature).
  Dim lngLeftNum As Long, lngRightNumber As Long
  lngLeftNum = getInterpolatedHouseNumber(lngFrom_Left_HouseNum, lngTo_Left_HouseNum, dblDistAlongCurve)
  lngRightNumber = getInterpolatedHouseNumber(lngFrom_Right_HouseNum, lngTo_Right_HouseNum,
dblDistAlongCurve)
  ' the following 10 lines set the TO_LEFT and TO_RIGHT numbers of the first feature.
  If lngFrom_Left_HouseNum = lngTo_Left_HouseNum Then ' if parent had no range of house numbers.
    pNewFeature1.Value(leftToFI) = lngFrom_Left_HouseNum
  Else
    pNewFeature1.Value(leftToFI) = lngLeftNum
```

```
  End If
  If lngFrom_Right_HouseNum = lngTo_Right_HouseNum Then
    pNewFeature1.Value(rightToFI) = lngFrom_Right_HouseNum
  Else
    pNewFeature1.Value(rightToFI) = lngRightNumber
  End If
  ' the following lines set the FROM_LEFT and FROM_RIGHT numbers of the second feature.
  If lngFrom_Left_HouseNum < lngTo_Left_HouseNum Then
    If mixedParity = False Then
      pNewFeature2.Value(leftFromFI) = pNewFeature1.Value(leftToFI) + 2
    Else
      pNewFeature2.Value(leftFromFI) = pNewFeature1.Value(leftToFI) + 1
    End If
  ElseIf lngFrom_Left_HouseNum = lngTo_Left_HouseNum Then ' if parent had no range of house numbers.
    pNewFeature2.Value(leftFromFI) = lngFrom_Left_HouseNum
  Else ' if house numbers run opposite to the polyline's digitized direction.
    If mixedParity = False Then
      pNewFeature2.Value(leftFromFI) = pNewFeature1.Value(leftToFI) - 2
    Else
      pNewFeature2.Value(leftFromFI) = pNewFeature1.Value(leftToFI) - 1
    End If
  End If
  If lngFrom_Right_HouseNum < lngTo_Right_HouseNum Then
    If mixedParity = False Then
      pNewFeature2.Value(rightFromFI) = pNewFeature1.Value(rightToFI) + 2
    Else
      pNewFeature2.Value(rightFromFI) = pNewFeature1.Value(rightToFI) + 1
    End If
  ElseIf lngFrom_Right_HouseNum = lngTo_Right_HouseNum Then
    pNewFeature2.Value(rightFromFI) = lngFrom_Right_HouseNum
  Else ' if house numbers run opposite to the polyline's digitized direction.
    If mixedParity = False Then
      pNewFeature2.Value(rightFromFI) = pNewFeature1.Value(rightToFI) - 2
    Else
      pNewFeature2.Value(rightFromFI) = pNewFeature1.Value(rightToFI) - 1
    End If
  End If
  pNewFeature1.Store
  pNewFeature2.Store
  m_pEditor.StopOperation "Split house numbers"
  On Error GoTo AfterStopOperation
  ' partially refresh the screen.
  Exit Sub
EH:
  MsgBox Err.Number & Chr(13) & Err.Description, vbCritical, "Can not Split"
  Exit Sub
ErrorAfterStartOperation:
  MsgBox Err.Number & Chr(13) & Err.Description, vbCritical, "Can not Split"
  m_pEditor.AbortOperation ' do not do Store any of the edits unless all are successful.
  Exit Sub
AfterStopOperation:
  MsgBox Err.Number & Chr(13) & Err.Description & Chr(13) & _
      "The line was split, but there was a problem with refreshing the screen.", _
      vbCritical, "Refresh problem"
  Exit Sub
End Sub



Function TryToGetValidHouseNum(strHouseNum As Variant) As Long
  ' attempt to get a valid Long Interger from the supplied candidate.
  ' returns -1 if not possible.
  On Error GoTo EH
  TryToGetValidHouseNum = -1
  ' do not allow candidates containing only blank spaces.
```

```
  If Trim(strHouseNum) = "" Then
    Exit Function
  End If
  strHouseNum = Trim(strHouseNum) ' strip away any blank spaces.
  ' drop non-numeric characters, if present.
  Dim lngHouseNumber As Long
  If IsNumeric(strHouseNum) Then
    lngHouseNumber = CLng(strHouseNum)
  Else
    Dim strNewCandidate As String
    Dim i As Integer
    Dim oneChar As String
    For i = 1 To Len(strHouseNum)
      oneChar = Mid(strHouseNum, i, 1)
      If IsNumeric(oneChar) Then
        strNewCandidate = strNewCandidate & oneChar
      End If
    Next
    If Len(strNewCandidate) = 0 Then ' if none of the characters were numbers.
      Exit Function
    End If
    lngHouseNumber = CLng(strNewCandidate)
  End If
  ' lngHouseNumber will be used by the main code.
  TryToGetValidHouseNum = lngHouseNumber
  Exit Function
EH:
  MsgBox Err.Number & Chr(13) & Err.Description, vbCritical, "Error in TryToGetValidHouseNum"
  Exit Function
End Function



Public Function getInterpolatedHouseNumber(lngFrom As Long, lngTo As Long, dblDist As Double) As Long
  ' interpolate the next lowest whole number given lngFrom and lngTo. Makes sure it is on the
  ' correct side of the street if MixedParity is False.
  ' start by returning the raw (Double) interpolated house number.
  On Error GoTo EH
  Dim dblHouseNum As Double
  Dim lngNextLowestHouseNumber As Long
  Dim mixedParity As Boolean
  mixedParity = False
  If lngFrom < lngTo Then
    dblHouseNum = ((lngTo - lngFrom) * dblDist) + lngFrom
    lngNextLowestHouseNumber = Int(dblHouseNum) ' Int will return a Long.
  Else ' if house numbers run opposite to line's digitized direction.
    dblHouseNum = lngFrom - ((lngFrom - lngTo) * dblDist)
    lngNextLowestHouseNumber = Int(dblHouseNum) + 1 ' Int will return a Long.
  End If
  If Not mixedParity Then ' the default.
    ' make sure the interpolated number is on the correct side of the street.
    Dim blnFromIsEven As Boolean, blnCandidateNumberIsEven As Boolean
    blnFromIsEven = isEven(lngFrom)
    blnCandidateNumberIsEven = isEven(lngNextLowestHouseNumber)
    If blnFromIsEven <> blnCandidateNumberIsEven Then
      If lngFrom < lngTo Then
        lngNextLowestHouseNumber = lngNextLowestHouseNumber - 1
      Else
        lngNextLowestHouseNumber = lngNextLowestHouseNumber + 1
      End If
    End If
  End If
  getInterpolatedHouseNumber = lngNextLowestHouseNumber
  Exit Function
EH:
```

```
  MsgBox Err.Number & Chr(13) & Err.Description, vbCritical, "Error in getInterpolatedHouseNumber"
  Exit Function
End Function


Function isEven(num As Long) As Boolean
  ' returns True if num is even.
  isEven = False
  If num Mod 2 = 0 Then
    isEven = True
  End If
End Function
```